CS190C Lec2

From Seq2Seq to Transformer

Overview

- What is Seq2Seq
- Seq2Seq with RNN
- Seq2Seq with Transformer
    - Attention
    - Position Embedding
    - Feed-Forward Network
    - Residual Connection
    - The Whole Transformer
- Thinking

PART1: What is Seq2Seq

Examples

- Given a Chinese sentence, translate to English $\Rightarrow$ Machine translation
- Given a long paragraph, conclude to a short sentence $\Rightarrow$ Abstract writing
- Given an academic report, convert to easy-to-understand article $\Rightarrow$ Style transfer

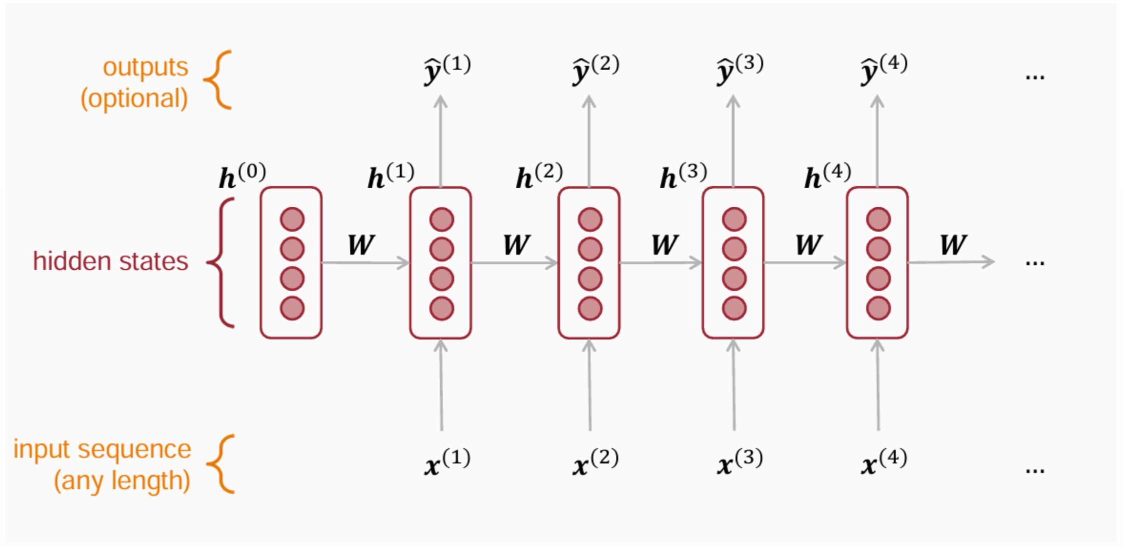
Conclusion

Seq2seq is a kind of language task to input a sequence and output a sequence according to certain demand.

PART2: Seq2Seq with RNN

Review

- At each time step, receive a word and encode it.
- Mix the encoded word and former text information.
- At each time step, the information (embedding) of text will be enriched once.
- At last time step, the embedding of text will contain the whole information of all words.

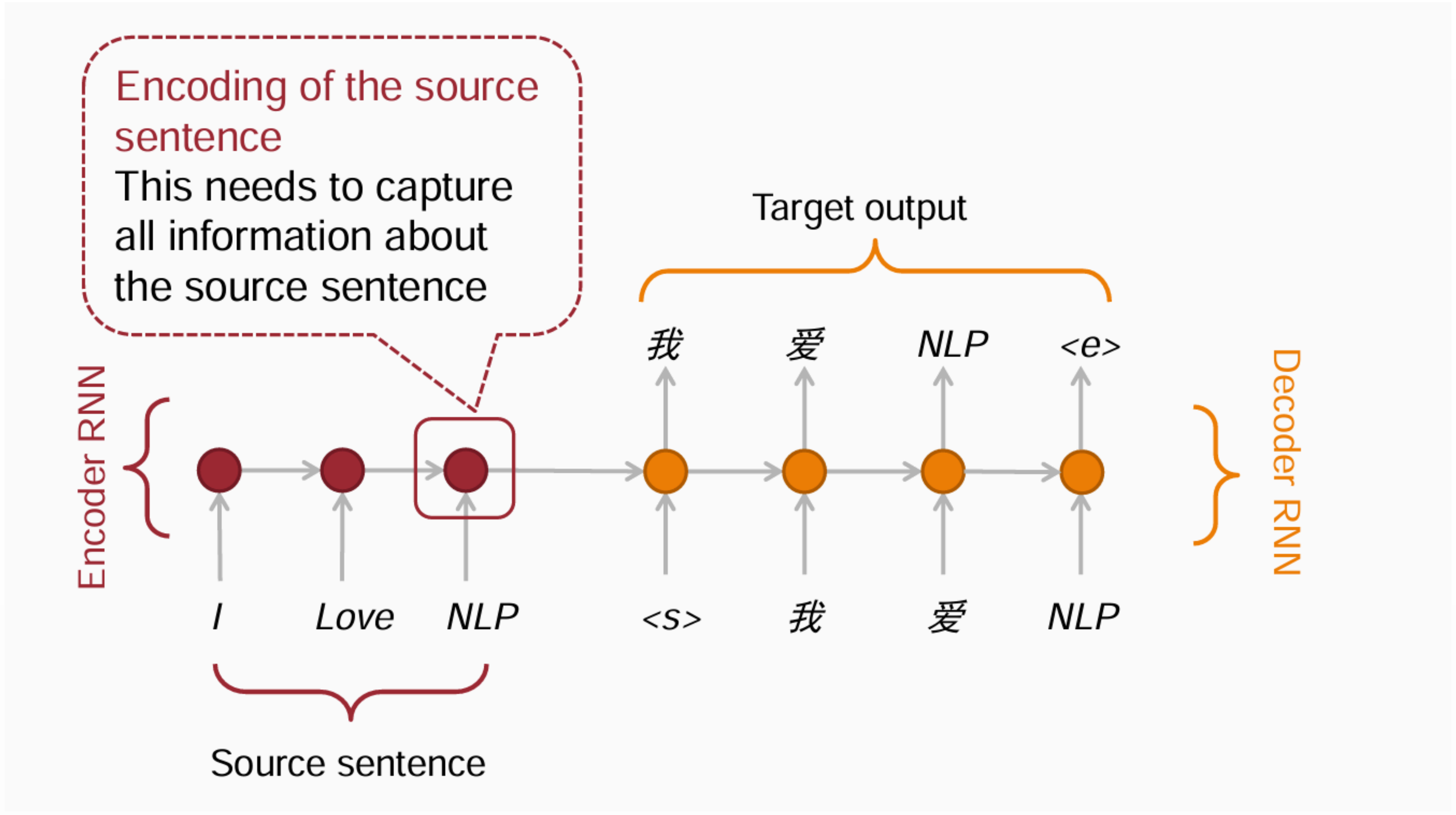

outputs (optional)
hidden states
input sequence (any length)
$\hat{y}^{(1)}$ $\hat{y}^{(2)}$ $\hat{y}^{(3)}$ $\hat{y}^{(4)}$ ...
$h^{(0)}$ $h^{(1)}$ $h^{(2)}$ $h^{(3)}$ $h^{(4)}$
W W W W W ...
$x^{(1)}$ $x^{(2)}$ $x^{(3)}$ $x^{(4)}$ ...

A naive idea

- The embedding of text will contain the whole information of all words at last time……
- Input a sequence and use RNN to process it……
- The last text embedding h^T is hoped to carry all information of it.
- We can further decode output sequence based only on h^T.
- We use another RNN to decode, right after the first RNN.

Seq2Seq with RNN

- Encoder and Decoder.


Encoding of the source sentence
This needs to capture all information about the source sentence
Target output
我 爱 NLP <e>
Encoder RNN
Decoder RNN
I Love NLP
<s> 我 爱 NLP
Source sentence

Pros and Cons?

- Easy and intuitive.
- RNN: Gradient Problems.

If the sequence is too long, the model's performance is not so good——not only gradient problems……

- Encode everything into a single vector without proper weights.
- Forward:$h^t = \sigma(W_h h^{t-1} + W_e e^t + b_1)$, remote information will receive significantly more changes——Although they may be important

`The writer of the books (is/are?)` $\Rightarrow$ `books` is closer and `writer` is farther.

PART3: Seq2Seq with Transformer

https://arxiv.org/abs/1706.03762

PART3.1：Attention

How to encode with proper weight?

```
The boy who is picking apples (is/are)?
```

For this word, it should find words whose properties has strong correlationship with its "problems".

- How to describe properties of each words' "problems"?
- How to describe properties of each words themselves?
- There's no doubt that: If "problems" and "properties" has strong relationship, it will cause significantly high weight, like obvious attention.
- If a word find another word significantly satisfied the requirements above, how to describe the information the word provide?

How to encode with proper weight?

- All above should rely on the words' embeddings.

If a word has embedding $x \in R^d$, each x_i $(1 \leq i \leq d)$ may stands for the significance level of a certain semantic feature......

- The whole semantic of a word can be seen as conbination of all semantic features and their significance level
- Each semantic feature may represent certain kind of problems, properties and information

How to encode with proper weight?

- For all d semantic features, we hope to train a matrix $W^Q \in R^{d*d_{qk}}$, which means: Certain semantic feature will cause a "problem", whose embedding $\in R^{d_{qk}}$
- Similarly, we hope to train a matrix $W^K \in R^{d*d_{qk}}$, which means: Certain semantic feature will represent a property, whose embedding $\in R^{d_{qk}}$.
- Similarly, we also hope to train a matrix $W^V \in R^{d*d_v}$, which means: Certain semantic feature will feedback certain kind of information, whose embedding $\in R^{d_v}$.

How to encode with proper weight?

So, how do we calculate the "problem" of a word, which is the combination of semantic features?

Recall: Certain value of x_i means the significance level of i-th semantic features.

$W^Q = [w_1^q w_2^q \ldots\ldots w_d^q]^T$.
$q = x_1 w_1^q + x_2 w_2^q + \ldots\ldots + x_d w_d^q$
$\Rightarrow q = xW^Q$, which is the "problem" of a word.

Similarly:$k = xW^K, v = xW^V$

$x \Rightarrow q, k, v$

Scaling Dot Product Attention (SDPA)

So, if word i with "problem" q_i, word j's weight score with word i is $q_i k_j^T$. Use softmax to normalize all words weight with word i to get weight distribution, which is the weight of sum v_j feedback to word i.

Write to matrix formular:

$$\text{Attention}_i = \text{Softmax}(\frac{QK^T}{\sqrt{d_{qk}}})V.$$

Tips: $\sqrt{d_{qk}}$ is the scaling factor to ensure Softmax logits not too large or too small, thus ensure the distribution not degenerate to one-hot or uniform.

Multihead Attention (MHA)

Does a semantic feature have only one "problem", only one "property"?

Maybe we can repeat SDPA in H channels, each channel stands for certain kind of "problem" and "property" pair. So word i has: h_1,h_2,......,h_H ($h_j = \mathrm{Attention}_i^j$).

Final $\mathrm{Attention}_i = \mathrm{Concat}(h_1, h_2, \ldots\ldots, h_H)W^O$

PART3.2： Position Embeddings

A Problem

- In real-world contexts, if two sentences have same word but different orders, they almost certainly has different meanings. `I study LLM` `LLM study I`
- But what in MHA?

Only word embeddings and their q, k, v matters, do not include position.

$\Rightarrow$ We should add some "tag" for each word, representing their position in the sentence. Thus it will have an impact on MHA.

Sine position encoding in Transformer 2017

We should bring a regular change to word embeddings x, but how?

- Actually the change should be a function of position.

Is it enough?

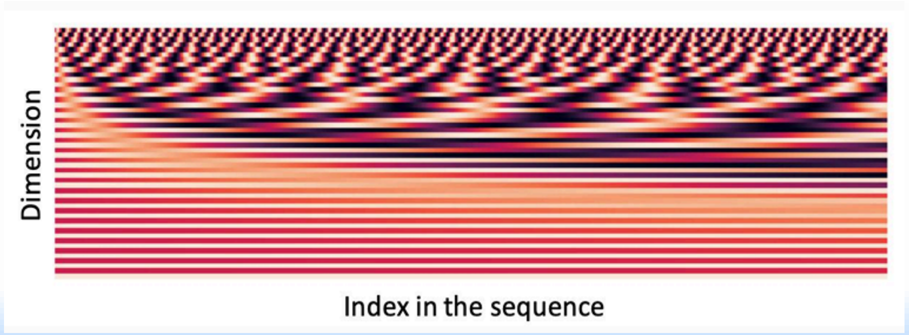
- Some kind of semantics may be sensitive to position change, but some are not.
- So the change should also be a function of dimension of word embeddings.
- Also, the function should be one-to-one mapped with position and dimension.

Sine position encoding in Transformer 2017

Let p_i be the position embedding at position i with dimension d.

Then $p_j = \begin{cases} \sin(pos \cdot \omega_j), & \text{if } j \text{ is even} \\ \cos(pos \cdot \omega_{j-1}), & \text{if } j \text{ is odd} \end{cases}, \omega_d = \frac{1}{10000^{j/d}}$

- Period$=\frac{2\pi}{\text{pos}} 10000^{j/d}$
- For different position, there will be a different embedding function $\Rightarrow$ one-to-one mapped.
- For different dimension: the period will change significantly from 2π to 20000π, standing for different sensitivity of different kind of semantics.



Pros and cons?

- It may also represent some relative position difference in a way.

$$\begin{bmatrix}\sin((pos+k)\omega_j)\\ \cos((pos+k)\omega_j)\end{bmatrix} = \begin{bmatrix}\cos(\omega_j k) & \sin(\omega_j k)\\ -\sin(\omega_j k) & \cos(\omega_j k)\end{bmatrix}\begin{bmatrix}\sin(pos\cdot\omega_j)\\ \cos(pos\cdot\omega_j)\end{bmatrix} = M_{k,j}\begin{bmatrix}\sin(pos\cdot\omega_j)\\ \cos(pos\cdot\omega_j)\end{bmatrix}$$

$$\Rightarrow p_{pos+k} = M_k \cdot p_{pos}\ (M_k = \mathrm{diag}(M_{k,1}, M_{k,2}, \ldots, M_{k,d/2}))$$

And in natural language, difference in semantic caused by position is usually relative position instead of absolute position.

Pros and cons?

- But sine position encoding is still absolute position.

For calculation of q_i and k_j:

$$q_i k_j^T = (x_i + p_i) W_Q W_K^T (x_j + p_j)^T = x_i W_Q W_K^T x_j^T + p_i W_Q W_K^T x_j^T + x_i W_Q W_K^T p_j^T + p_i W_Q W_K^T p_j^T$$

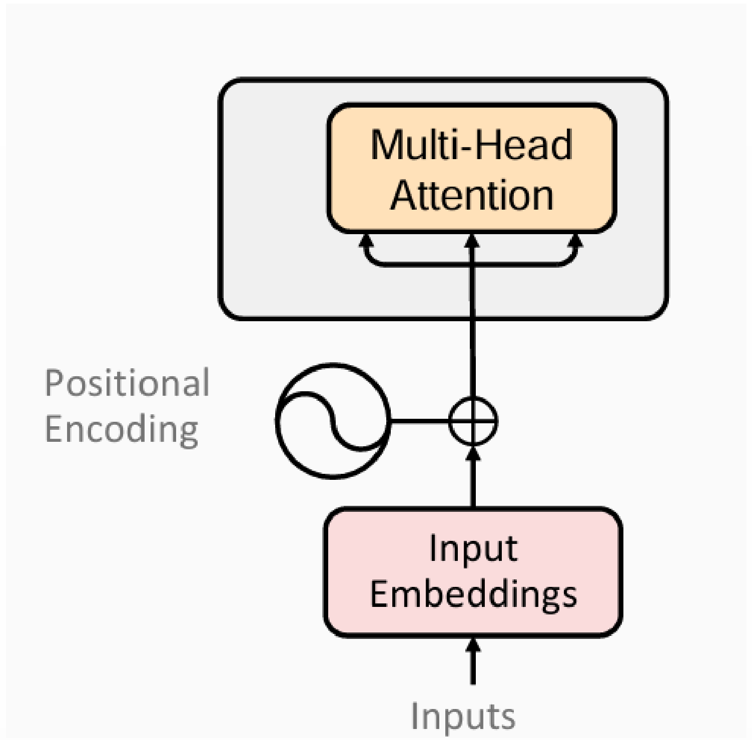
- For $p_i W_Q W_K^T p_j^T$, if $W_Q W_K^T$ does not satisfy the structure of M_k, it is related to absolute position.
- For $p_i W_Q W_K^T x_j^T$ and $x_i W_Q W_K^T p_j^T$, it is completely determined by absolute position.

What is the disadvantage of absolute position?

- Difference in semantic is usually not depend on it.
  - But it is different for attention calculation given a pair of tokens with the **same relative position** and **different absolute position**
  - The model may remember difference in abosulute position, but it is unnecessary.
- When the model faced very long sentences in reasoning after training......
  - The model don't know how to process such absolute position, because it hasn't be trained with it.
  - The outcomes will worsen.

So most LLMs now use `RoPE` to embbed position, which is a relative position embedding method (In Lec3).

The language model for encoding so far……



- Add position embeddings before enter the encoder block.
- Enter the encoder block, using Multi-Head Attention to enrich the word embeddings using the information of text.
- We can use multiple layers of blocks to enrich it multiple times.

But there still exists some problems……

Problems

- MHA is essentially a linear operation. So it cannot process non-linear information.
- Moreover, word embeddings can only absorb information from the text.

```
I twisted the door handle, and the door ( )
```

To answer `opened`, the model should learn:

- Common sence of physics: Twisting will make it "open" instead of "broken".
- Common sence of life: We use handle to open it instead of threshold.
- Other "common sences" outside the text and beyond MHA.

How can we fix it?

PART3.3: Feed-Forward Network

Dimension ascent

Suppose there are a great many of "global features", such as: "Is it a tool to open something?"

How can we calculate the relationship between the word embedding and this feature?
$\Rightarrow$ We still use dot product. That is: xk_i^T is the relationship of x and feature i.

Moreover, we add an activation function σ to it, such as ReLU: $\mathrm{ReLU}(x) = \max(x, 0)$

We have: $x_{ff} = \sigma(xW_1)$, meaning "the relationship between word embedding and different features"

https://arxiv.org/abs/2012.14913

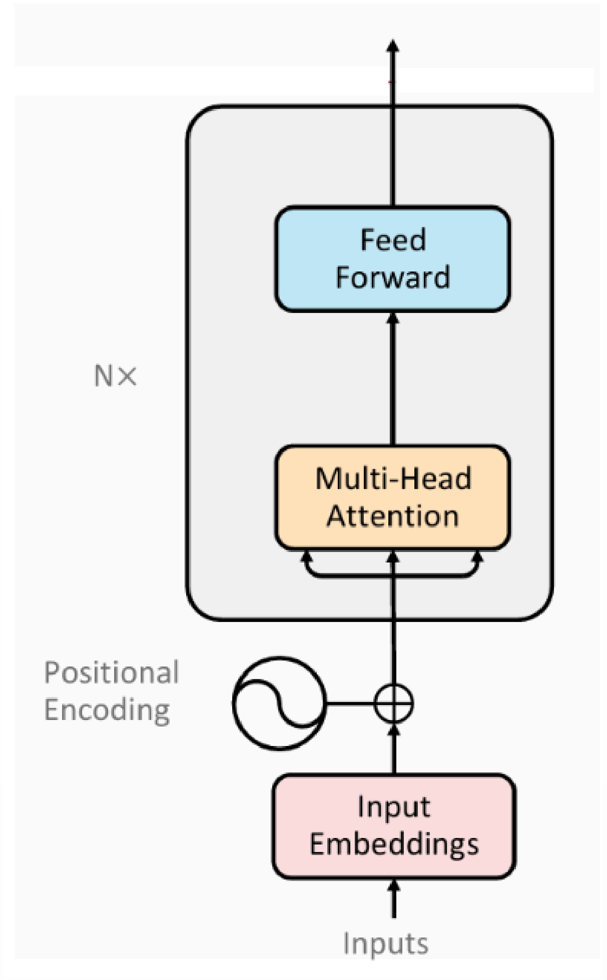
Dimension descent

We return the dimension of x to the original state.

$x_{output} = x_{ff}W_2$. That is: projection of feature vector with high dimension to the original dimension.

FFN: $x_{output} = \sigma(xW_1)W_2$. Dimension of x_{ff} is usually $4d$.

The language model for encoding so far……



- Using self-attention to capture contextual information from the text.
- Applying Feed-Forward Networks (FFN) to extract global feature information.
- Repeat the above steps for N layers to deepen representation.

However, as N increases, training becomes more difficult. It is said that for such a kind of model, we can only effectively train 20~30 layers.

How can we address this challenge?

PART3.4: Residual Connection

Perspective of forward: A large number of layers?

- Suppose for a really large layer number N, we actually need several layers to be expressive enough.
- So it means: If N is very large, some layers can act as identity transformation to maintain the information.
- However, it is hard to train an "identity transformation", that is matrix I using `MHA` and `FFN`
- So large N may leads to degenerate on the contrary.

Perspective of forward: A large number of layers?

Question: It is hard to train I for a layer, but is it hard to train 0 ?

$\Rightarrow$ We have $L2 - \mathrm{Norm}$: parameters trend to 0.

- So why don't we learn the parameter based on I? That is: only learn difference of information, not absolute information.
- $y_l = \mathrm{Attn}_l(x_l) \Rightarrow y_l = \mathrm{Attn}_l(x_l) + x_l$
- $x_{l+1} = \mathrm{FFN}_l(y_l) \Rightarrow x_{l+1} = \mathrm{FFN}_l(y_l) + y_l$

It can significantly avoid degenerate of too large N.

Perspective of backward: Chain rule?

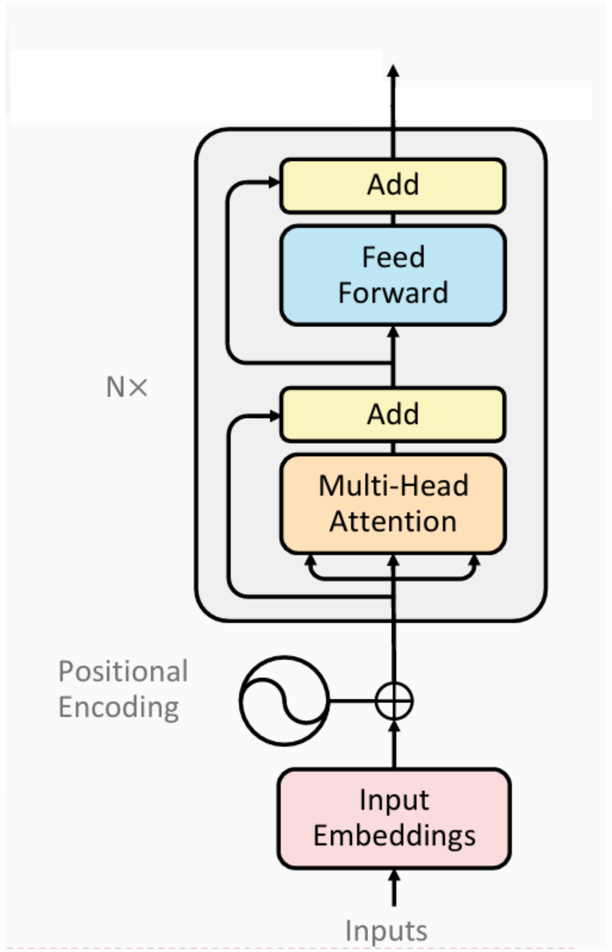
$$\frac{\partial L}{\partial x_l} = \frac{\partial L}{\partial x_{l+1}} \frac{\partial x_{l+1}}{\partial y_l} \frac{\partial y_l}{\partial x_l} = \frac{\partial L}{\partial x_{l+1}} \cdot \mathbf{Attn}_l' \cdot \mathbf{FFN}_l'$$

That is: For large N, we need to multiply many terms to update parameters of bottom layers.

Familiar? It is the same as RNN.

The most serious problems is vanishing gradient, which leads to almost not learnable parameters of bottom layers. For example, if a layer is fully learned, it cause a term almost 0.

Perspective of backward: Chain rule?



If we add residual connection?

- $\frac{\partial L}{\partial x_l} = \frac{\partial L}{\partial x_{l+1}} \frac{\partial x_{l+1}}{\partial y_l} \frac{\partial y_l}{\partial x_l} = \frac{\partial L}{\partial x_{l+1}} (I + \mathrm{Attn}_l')(I + \mathrm{FFN}_l')$
- Even if some modules has gradient approches to 0, the gradient can still effectively transmit to botton.

PART3.5: Layer Normalization

Estimate the scale of activations

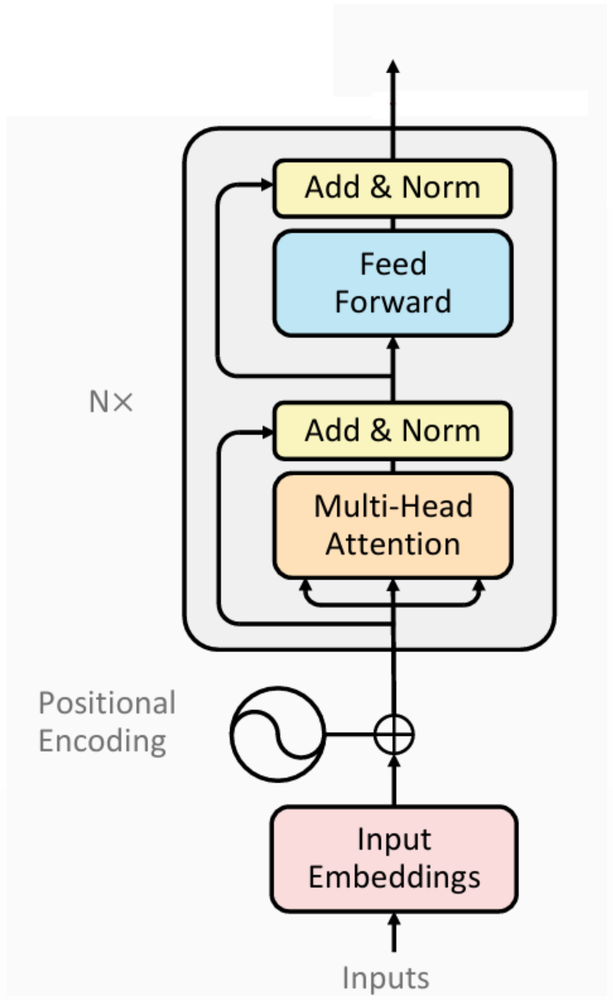
- For a certain tensor of activation with multiple elements, we consider value of elements as a distribution.
- The mean of distributions is usually approches to 0, but the variation differs. We usually use variation to measure order of magnitude of activations.
- In a residual network, what is the variation of activations of modules in different layers?

Estimate the scale of activations

In preliminary stage of training, parameters are initialized randomly. So activation of each module F is approximately independent of input of module.

- $\mathrm{Var}(x_2) = \mathrm{Var}(x_1 + F(x_1)) = \mathrm{Var}(x_1) + \mathrm{Var}(F(x_1)) = (1+\alpha)\mathrm{Var}(x_1)$
- $\mathrm{Var}(x_3) = \mathrm{Var}(x_2 + F(x_2)) = (1+\alpha)\mathrm{Var}(x_2) = (1+\alpha)^2\mathrm{Var}(x_1)$
- $\mathrm{Var}(x_n) = (1+\alpha)^{n-1}\mathrm{Var}(x_1)$
- Exploded!

Layer norm



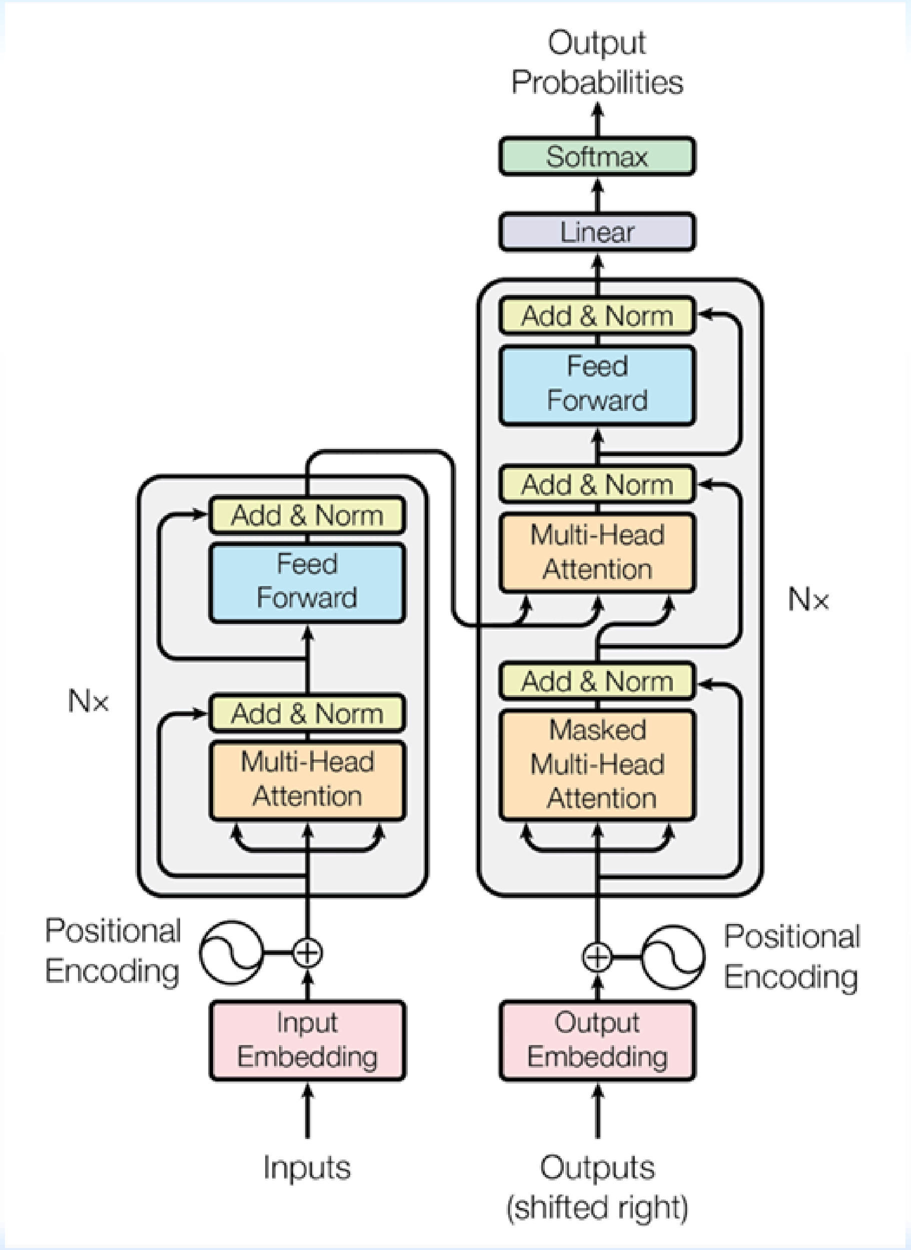
We should perform a normalization to activations in each module. That is:

$$\hat{x} = \frac{x - \mu}{\sigma}$$

It can reduce the explosion of activations.

In next lecture, we will discuss where to perform (post-norm & pre-norm), how to perform(LN & RMSNorm)

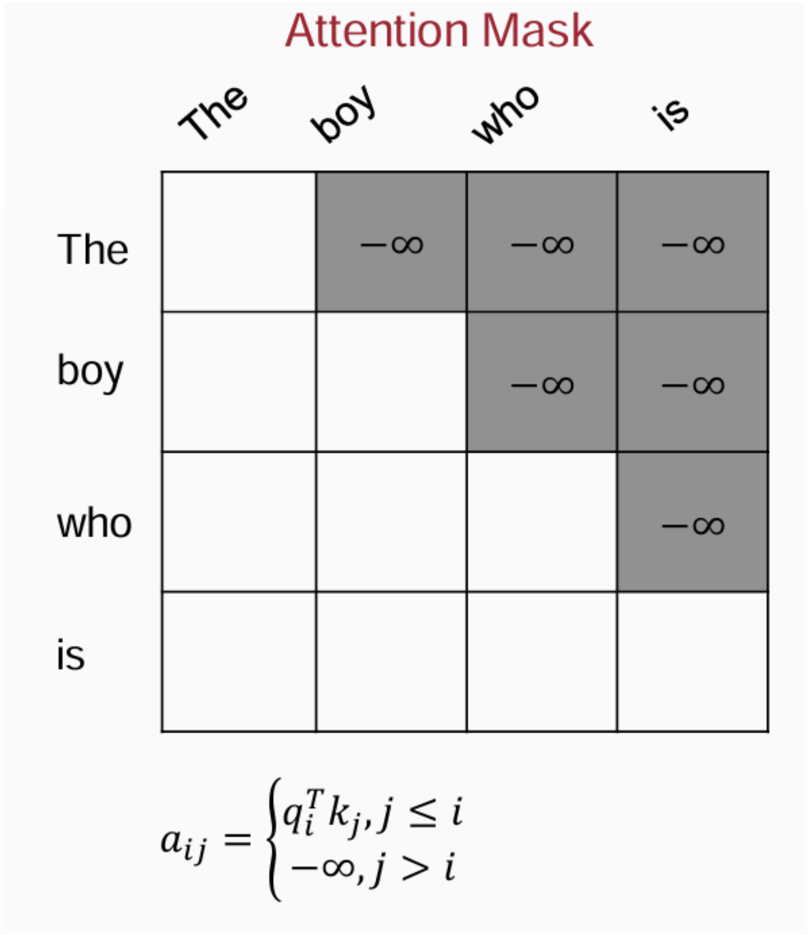
PART3.6: The Whole Transformer



Seq2Seq with Transformer

- Encoder
  - Use N layers to encode the input embeddings, the output of N-th layer is the encode outcomes.
- Decoder
  - Use a self-attention module to absorb the decoded information.
  - Use a cross-attention module to absorb the information of encoder.
- Linear
  - Project decoder from embedding to distributions over vocabulary words.

What is masked MHA?



$$a_{ij} = \begin{cases} q_i^T k_j, j \leq i \\ -\infty, j > i \end{cases}$$

For QK^T matrix:

- We add it an upper traingle mask, to make the upper part equals to $-\infty$.
- When we apply Softmax to it, it means: the upper part elements have weight 0.
- It means: when i-th word try to absorb information of other words, j-th word should contribute no information if $j > i$.

Why Masked MHA?

What's the training sentences look like?

$\Rightarrow$ `inputs` + `outputs`

- During training process: The model decoder try to output `seq_len` embeddings, that is the "sentence" decoded.
- The i-th position means: Absorbing information of input sentence(encoder output) and 1-th$\sim$ i-th decoder output, predict the embedding of $i+1$-th word.
- So we cannot make model know any information about words after position i, otherwise it is information leak.